PATENT OFFICE OF THE PEOPLE'S REPUBLIC OF CHINA
PUBLIC DESCRIPTION OF THE INVENTION PATENT APPLICATION
PUBLICATION NO. CN 1155700A

Int. Cl.[6]:	G 06 F 17/00
Filing No.:	96112824.0
Filing Date:	September 8, 1996
Publication Date:	July 30, 1997

COMPUTER SOFTWARE PROTECTING METHOD

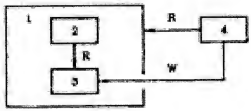
Inventors:	Yang Zhuping and Zhou Yueping
Applicants:	Zhou Yueping Mathematic Group, Dongwan Teacher Training College, Dongwan, Guangdong Province 511700

Number of pages of claims:	1
Number of pages of specifications:	5
Number of pages of figures:	1

[There are no amendments to this patent.]

Abstract

The present invention provides a computer software protecting method. It pertains to computer security, more particularly, to software development and protection. Based on the concept and idea of software fingerprint and software purity, the essential conditions for falsifying software and the sufficient conditions for not falsifying software are proposed. The fingerprint is checked to identify the purity to guarantee that the software itself can promptly determine that it is being falsified and can inhibit further falsification. The steps include adding protective code sequence (2) into software (1) and using tool (4) to extract and record software fingerprint (3) in advance. This method is characterized by the fact that a protective code sequence is generated during development of the software and the software fingerprint is a statistical characteristic. The present invention provides software with a self-immunization capability so that it is possible to effectively prevent spreading of a virus.



Claims

1. A type of computer software protecting method, wherein the software includes a protective code sequence and a software fingerprint is extracted and recorded in advance, said protective code sequence is executed when the software is run to compare the current software fingerprint with the recorded software fingerprint to identify the purity of the software itself and to inhibit further running of the software after it is falsified, characterized by the following facts:

said protective code sequence is generated as part of the software functional codes during development of the software;

said software fingerprint is a statistical characteristic; it is obtained by using the consecutive byte sequence of the entire or part of the content of a specific file in the software itself as data based on a calculation carried out according to a prescribed algorithm, and it is recorded in the form of data.

2. A type of computer software including a protective code sequence that can check the software fingerprint during execution, characterized by the fact that said protective code sequence is used to provide a function call for another software or generate a protective code sequence for another software.

3. The computer software protecting method described in Claim 1 characterized by the fact that said protective code sequence and the software fingerprint extracted in advance are generated by a program language complier or by the software development system.

4. The computer software protecting method described in Claim 1 characterized by the fact that said protective code sequence is included in the file that it directly protects.

5. The computer software protecting method described in Claim 1 characterized by the fact that the main body of said protective code sequence is outside the file that it directly protects.

6. The computer software protecting method described in Claim 1 characterized by the fact that the main body of said protective code sequence is included in the system running environment.

7. The computer software protecting method described in Claim 1 characterized by the fact that said software fingerprint extracted in advance is recorded in the same file where the protective code sequence is located.

8. The computer software protecting method described in Claim 1 characterized by the fact that said software fingerprint extracted in advance is recorded outside the file where the protective code sequence is located.

9. The computer software protecting method described in Claim 1 characterized by the fact that said software fingerprint extracted in advance includes the fingerprints extracted separately from two or more files of the software that are recorded collectively in one file.

10. The computer software described in Claim 2 characterized by the fact that it can generate a software fingerprint for other software.

Specification

The present invention provides a type of computer software protecting method. It pertains to computer security, more particularly, to software development and protection.

The harm caused by spreading of a computer virus is a serious problem faced by computer software protection. Most of the conventional protecting methods use a certain type of protection mechanism as the running environment to monitor and diagnose computer software. An example is the patented invention "stored information protecting mechanism" (Chinese Patent No. 90101742.6). This kind of method is difficult to promote due to diversification and continuous updating of system environment. It is also possible to use a certain special virus scanning software to scan the code files of other software to identify and eliminate viruses based on known characteristics of the viruses. An example is the program MSAV.EXE in DOS6.20 developed by Microsoft. Due to its limitations and the variations in viruses, this kind of method usually makes mistakes in detecting viruses or overlooks viruses. Another method is to establish a self-protecting mechanism. An example is the published invention "software self-protecting method for preventing computer virus" (publication No. CN1068205A). According to said invention, a program section attached to the original software is used as a protective shell to record the file length and the characteristic parameters of the running entry. Said program section is run prior to the original software to detect and eliminate a virus based on a comparison between the recorded and the current characteristic parameters. However, this invention has obvious defects. First, under current circumstances when software is circulated extensively and privacy software is everywhere, the "origin" of a software is doubtful. Therefore, it is risky to add a shell. Second, the characteristic parameters disclosed in this invention, that is, file length and running entry cannot fully reflect the characteristics of a file, which shakes the application foundation of said invention. The reason is that it is absolutely possible to design a virus that is parasitic in the file without changing the file length or running entry. Third, the measure adopted by this invention to eliminate a virus is actually based on the following assumption, that is, a virus is always attached to the end of a file. However, regretfully, this assumption is not valid.

A more general problem for software protection is to prevent falsification of software. Besides falsification by a virus, other kinds of falsifications include changing copyright claims, changing the language, and changing program codes. Accidental falsifications caused by random problems or physical damage to the memory also occur. All of said falsifications should be inhibited since they are opposed to the will of the copywriter.

The objective of the present invention is to provide a common and effective protecting method for preventing falsification of computer software.

The entity of computer software consists of saving files. Therefore, protecting software means actually protecting its files. The constitution of software includes at least one program file and one or more data files. The key to protecting software is to protect the program files and the important data files.

First, the present invention introduces the concept of software fingerprint. More specifically, a software fingerprint is its file fingerprint. It is an intrinsic static characteristic of a consecutive byte data sequence that uses simple values known as characteristic values or small data blocks to fully represent or reflect the file content. Regarding fingerprint characteristic, there is no difference between the program file and data file of software. Any change to any byte in the file will lead to a change in the fingerprint characteristic of the file. Obviously, the fingerprint characteristic is a statistical characteristic, which is calculated according to a prescribed algorithm. There are many kinds of algorithms. Different algorithms may have different practical application effects. A cumulative calculation of the byte data of a file is a simple algorithm. The accumulated sum is known as the check sum, which can be used as the characteristic value of a type of file fingerprint (although it is not refined yet). The application example of the present invention will provide a highly-effective and practical algorithm.

The present invention also introduces the concept of file purity. Software is pure if its program files and important data files are pure. A file is pure if its content is exactly the same as the original content. A pure file is not falsified, while a falsified file is not pure. A thorough method for guaranteeing the purity of a file is to compare the content of the current file with the content of the original file byte by byte. However, such a primitive method is impractical. The present invention adopts a quantization method, which checks whether a file is pure by checking and comparing the current fingerprint characteristic of the file with the original fingerprint characteristic. The file is pure if the characteristic values are the same. Otherwise, the file is impure. The integrity and correctness of the software can be guaranteed by protecting the purity of the software.

Based on the concept and idea of software fingerprint and software purity, the following conclusion can be drawn: the essential condition for falsifying software is that its current fingerprint characteristic is different from the original fingerprint characteristic; the sufficient

condition for not falsifying software is that its current fingerprint characteristic is the same as the original fingerprint characteristic. This is the basis of the present invention.

It is not alarming if software is falsified because there is a method for resuming the software. The alarming part is for software to be run continuously after it has been falsified because the resulting loss can be immeasurable. The effect targeted by the present invention is not to try to guarantee that software will not be falsified. Instead, the objective of the present invention is to guarantee that software itself can promptly find any existing falsification in itself and can prevent spreading of the falsification behavior.

The method of the present invention includes the following two basic steps:

The first step is to add a protective code sequence to the software. More specifically, this is added in a program file. When executed, the protective code sequence will achieve the following four functions:

1. Identify the exact directory path of the main file and other files as protection objects.

2. Calculate the current (prior to running) characteristic value of the file fingerprint according to a prescribed algorithm.

3. Compare the current characteristic value of the file fingerprint with the original characteristic value that has been extracted and recorded.

4. If the characteristic value of the file fingerprint has changed, it means that the file has been falsified, and the situation is handled according to a predetermined rejection strategy.

The basic use of the protective code sequence is to monitor and protect the program file where it is located. It can also be used to monitor and protect other files, especially, important data files. It is necessary to identify the file path in order to prepare for checking of the file fingerprint. The rejection strategy should be easily understood and prepared, such as a warning with a sound or message, automatic exit, or as determined by the operator.

A protective code sequence can be "added" from outside. However, the present invention recommends adoption of the protecting function as one of the essential functions of the software. Therefore, the protective code sequence should be encoded integrally as part of the software code during development of the software.

The protective code sequence is present in a program file. If software has plural program files, the main body of the protective code sequence can be stored collectively in one file, while the protective code sequence in the other program files is simplified to an entry command sequence for calling the main body of the protective code sequence. The main body of the protective code sequence can also be standardized so that it can be included in an operating system, Chinese character system, database management system, or other running environments.

The static position, that is, the storage position of an added protective code sequence in a file, is not important. It is important that the dynamic position of the protective code sequence be

at the activation position on the program running path. It should be arranged that the protective code sequence is executed at the beginning of program running or prior to an important processing operation. It is also possible to execute the protective code sequence after the program has been run for a certain period of time (for example, when the program is about to end). In order to achieve a better protection effect, it is possible to activate and execute the protective code sequence on the program running path multiple times.

The simplest mechanism for activating the protective code sequence is sequential execution. Usually, a function or subroutine is called. It is also possible to call an interrupt function or even install a special program for execution.

The second step is to extract and record the software fingerprint in advance. After it has been confirmed that software is integral and correct, for example, after software encoding and testing has been finished during the development period of software, an external software tool is used to extract the fingerprint of the objective software and record it in a file in the objective software.

The software fingerprint is extracted by separately extracting fingerprints of the program file and important data files to protect in the software. In other words, a statistical calculation is carried out according to a prescribed algorithm with respect to the consecutive byte data sequence within the prescribed protection range of the file document to obtain characteristic values. Said prescribed protection range is usually the entire length of a file. If some program files need to write back themselves normally during running, the write back region of the file is considered as a variation region and will be ignored. Under special circumstances, said prescribed protection range can be consecutive sections. Said prescribed protection range is kept consistent with the file range monitored and protected by the protective code sequence. The operation semanteme of the command codes in the program file is completely ignored and is used as pure data to extract the software fingerprint. The algorithm adopted is the same or complementary with the algorithm in the protective codes. Their calculation results, that is, the file fingerprint characteristic values, are comparable.

The extracted software fingerprint characteristic value is recorded in a file in the software. The detailed position is predetermined such that the value can be read correctly by the protective code sequence during execution. For software having a single program file, the fingerprint characteristic value is recorded in that file. For software having plural files, the fingerprint characteristic values of each protection object file can be recorded in different files or collectively in one file.

The aforementioned two steps are not difficult to realize. It is absolutely possible to include them in the software development process. In order to increase the efficiency and for the purpose of regulation, the main body of the protective code sequence can be standardized and

stored in a source program library or target program library to be shared. Also, a tool used for extracting and recording a software fingerprint can be developed. It is even possible to design a new compiling program and development system equipped with such function, which can automatically generate software protecting codes and a software fingerprint.

In real life, falsification of software is inevitable. The effect of the present invention is that the falsification can be accurately identified when software is run for the first time after falsification and continuous running of the software can be prevented to avoid further loss. This also allows the user to take time to adopt the resuming measures. The present invention does not try to automatically resume directly from the falsified software itself because operation resumed in this way is not always reliable and is sometimes even harmful. In fact, the most direct, simple, and effective method for resuming falsified software is to use a pure software copy to cover it. With the aid of the present invention, the purity of software can always be guaranteed.

Application of the present invention can provide software with self-immunity so that spreading of a computer virus can be stopped effectively. In addition, the present invention can also help protect a software copyright and the benefits of software developers. The software development cost needed for implementing the present invention is very low, while the positive effect generated by the present invention is significant. If the various software developers bear the protection responsibilities for their software and all of the popular system software and application software can protect themselves, there will be no way for a computer virus to spread, so users can use their computers more safely.

Figure 1 shows the configuration of the present invention. Software 1 is a file set composed of at least one program file, usually, more than one program file and several (zero or one or more) data files. Protective code sequence 2 is stored in the program file of software 1 and is encoded integrally during the development period of software 1. A variation of protective code sequence 2 is a simplified command sequence used for calling the main body of the protective code sequence, while the main body of the protective code sequence is stored in another file or even included in the system environment wherein software 1 is run. Software fingerprint 3 is a group of statistical characteristic values calculated from a consecutive byte sequence in the prescribed protection range of each file as the protection object in software 1. External software tool 4 is used to extract the software fingerprint 3 of software 1 and is written into the file of software 1. It has characters marked on a straight line. R represents an object designated by a reader, while W represents an object designated by a writer.

Figure 2 shows the execution process of the protective code sequence in the present invention, wherein "extract the current file fingerprint" includes various essential operations, such as open file, read byte data, statistically calculate the fingerprint characteristic value according to the prescribed algorithm, and close file. "Compare file fingerprint" compares the

current fingerprint characteristic values of the file with the original recorded fingerprint characteristic values. "Normal return" means that the software is executed continuously. "Reject to execute" means that falsified software is disposed of according to a prescribed strategy.

In the following, an application example of the present invention will be explained. In this case, the present invention is applied to a single program file in a DOS environment. It should be easy to handle the case of plural program files and data files based on this application example.

A cyclic redundancy check (CRC) algorithm is used to extract the file fingerprint. A polynomial is generated, and a 16-bit CRC-CCITT is taken, that is, $X^{16}+X^{12}+X^5+1$. The 16-bit CRC value of the file byte data is used as the characteristic value of the file fingerprint. CRC technology has been used widely in the communication field. The theory and long-term practice have proven that its error checking effect is very ideal. Therefore, it is highly appropriate to use the CRC value as the file fingerprint.

The protective code sequence is added when designing the program. It is programmed in the form of a function. The first command of the main program immediately calls that function to execute the following operation.

1. Take the first parameter string from the parameters of the command line (the first byte is the line length) at the place that is 128 lines shifted from the program section prefix (PSP) and use it as the exact directory path of this program file.

2. Open the file and take the entire length of the file, read in all of the data, deduct the last word (2 bytes), calculate the CRC value, then close the file.

3. Compare the obtained CRC value with the last value (original CRC value) of the file.

4. Return normally if the two CRC values are equal. Otherwise, advise the user that the program has been falsified and then end the running.

A tool program with the file name CT.EXE is designed to complete the following functions.

1. Among the parameters of the command line (the first byte is the line length) at the place that is 128 lines shifted from the program section prefix (PSP), skip the first parameter string (that is, the directory path of the program itself) and take the second parameter string as the exact directory path of a given objective file.

2. Open the file and take the entire length of the file, read all of the data, calculate the CRC value.

3. Move the file indicator to the end, write in the CRC value to increase the file length by 2 bytes.

4. Close the file.

Said program section prefix (PSP) is established when running the loading program of the DOS system. The command line parameters starting from the place that is 128 lines shifted from said program section prefix are also filled in by the system and can be read by the program during running.

Assuming that a new program including the protective code sequence has the file name PG.EXE, said tool program CT.EXE can be used to execute the following at the DOS command line.

CT PG.EXE

In this way, the software fingerprint is extracted and recorded with respect to program PG.EXE in advance, and program PG.EXE is thus provided with a self-protecting function.

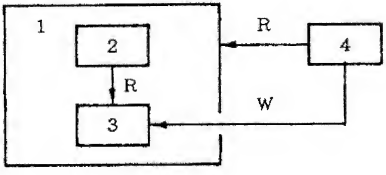


Figure 1

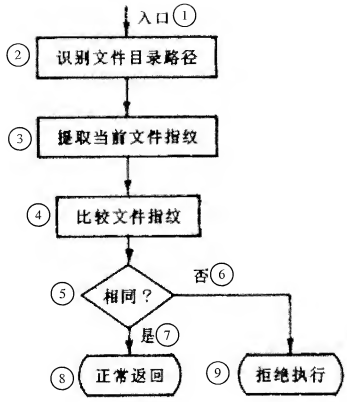


Figure 2

Key:
1 Entry
2 Identify the directory path of the file
3 Extract the current fingerprint of the file
4 Compare the file fingerprints
5 Same?
6 No
7 Yes
8 Normal return
9 Reject execution